CS 134: Networks

Prof. Yaron Singer Spring 2017

1 Overview

We've spent some time discussing networks in the context of information dissemniation—the HITS algorithm, influence maximization—and we're going to turn to a slightly different set of problems. Notice that in the first half of the course, we discussed structural properties of the networks. When we started thinking about influence maximization, this is the first time we started thinking about actually designing algorithms to ask questions about network data. Given a graph, how do we analyze it or manipulate it in a way that seems intuitive to us?

We now turn to a specific topic in the field of unsupervised learning known as *clustering.* With clustering, we can think about it both in the context of networks and separately from networks. Given some points in space, we might want to group similar points together. For instance, given a photograph, we could use clustering approaches to identify what pixels belong to the foreground of an image, and what pixels belong to the background. Given a set of rich data, we want to find k clusters that are somehow "representative" or most prominent in the data set. Our discussion of clustering is drawn largely from the textbook *Understanding Machine Learning: From Theory to Algorithms.*

Before we start thinking about algorithms, we can see that this problem is quite difficult.

- Imagine a set of points distributed uniformly along a grid. How might we approach clustering these points? We could cluster horizontally along the grid, or vertically along the grid. How do we know which clustering is better? The first thing about clustering that is not immediately apparent is the metric of success. **How do we define if a clustering is successful or unsuccessful?** In fact, the metric of success we define will determine the guarantees of the model.
- Imagine a set of points distributed into four concentrated clusters. However, let's say we want to cluster the data into two clusters—there's no "ground truth" about which clustering is better.

2 Clustering and Community Detection

In a network, we might want to apply clustering to the problem of "community detection." That is, given a graph $G = (V, E)$, how do we partition V into $C_1, C_2, C_3, \dots$ (where $C_i \cap C_j = \emptyset$ for all $i \neq j$ and $C_1 \cup C_2 \cup \dots = V$) such that each C_i seems to describe a cohesive "community" within G?

3 Clustering via k-means

The k-means algorithm is among the most popular clustering algorithms. It belongs to a clustering approach that begins by defining a cost function over a set of possible clusterings (an "objective function") and attempts to find a clustering that minimizes this function.

In k-means, the data is partitioned into disjoint sets $C_1, \dots, C_k$ where each C_i is represented by a centroid μ_i. The k-means objective function measures the squared distance between each point in X to the centroid of the cluster it belongs to. Here, X refers to the "data" that we want to cluster, but in the case of social networks in particular, X will become a set of vertices V. Assuming that X is embedded in some larger metric space X' with distance function d, such that $X \subseteq X'$, the centroid of C_i is defined as follows:

$$\mu_i = \mu(C_i) = \operatorname*{arg\,min}_{\mu \in X'} \sum_{x \in C_i} d(x, \mu)^2$$

The centroid of C_i is simply a value that, for each cluster, minimizes $d(x, \mu)^2$ for all x in that cluster. Let $\mu_i = \mu(C_i)$ for clarity. The k-means objective function is defined as follows:

$$G_{k-means}((X, d), (C_1, \dots, C_k)) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i)^2$$

Using the centroid definition, this can be rewritten as:

$$G_{k-means}((X, d), (C_1, \dots, C_k)) = \min_{\mu_i, \dots, \mu_k \in X'} \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i)^2$$

However, it turns out that actually finding the optimal solution that minimizes the k-means objective function is NP-hard–in fact, even approximating the optimal solution within a constant factor is NP-hard. The k-means algorithm is therefore a heuristic algorithm.

The algorithm below describes the k-means algorithm with the Euclidean distance function $d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||$. The approach is to alternate between assigning each point in X to the cluster with the nearest centroid in the previous iteration and then recompute each cluster's centroid.

```
ALG 1: k-means Algorithm
input: $X \subset \mathbb{R}^n$; number of clusters $k$
initialize: Randomly choose initial centroids $\mu_1, \dots, \mu_k$.
repeat until convergence
  $\forall i \in [k]$ set $C_i = \{x \in X : i = \arg\min_j ||\mathbf{x} - \mu_\mathbf{j}||\}$
  (break ties in some arbitrary manner)
  $\forall i \in [k]$ update $\mu_\mathbf{i} = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$
return All $C_i$
```

Note that the algorithm need not square norms when assigning vertices to clusters because squaring preserves order.

Lemma. *Each iteration of k-means algorithm does not increase the k-means objective.*

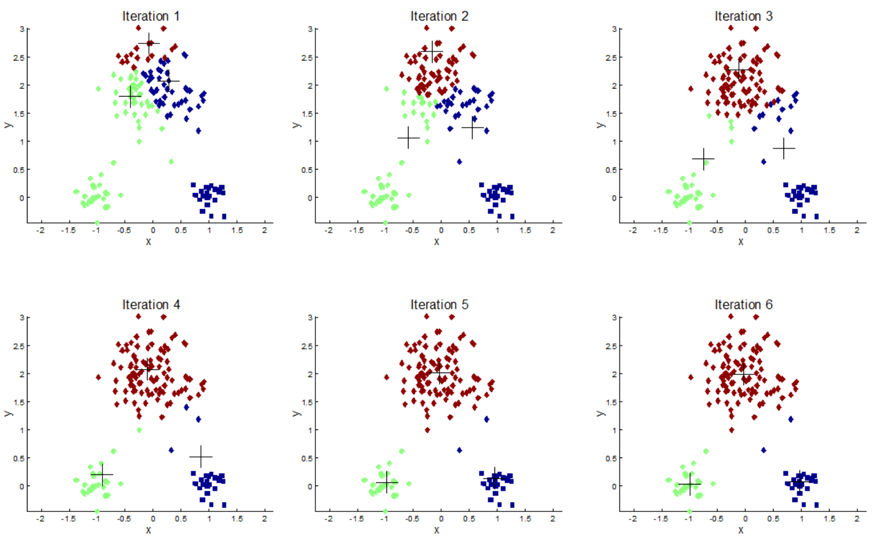

Figure 1: source: https://apandre.files.wordpress.com/2011/08/kmeansclustering.jpg

The k-means objective is defined (along the Euclidian norm) as the following:

$$F(G, \mathbf{C}_k) = \min \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$$

At each iteration t, let $C_{1,t}, \ldots, C_{k,t}$ be the clusters at time step t, and $\mu_{1,t}, \ldots, \mu_{k,t}$ be the centroids at each time step t. When we look at the objective at stage t, it is equal to

$$F(G, \mathbf{C}_{k,t}) = \min \sum_{i=1}^{k} \sum_{x \in C_{i,t}} ||x - \mu_{i,t}||^2$$

But we updated the centroids, which must be better than the centroids from the previous iteration. So the above expression must be less than or equal to:

$$\min \sum_{i=1}^{k} \sum_{x \in C_{i,t}} ||x - \mu_{i,t-1}||^2$$

But we had also updated the clusters to improve squared distances to those centroids, so the above expression must be less than or equal to:

$$\min \sum_{i=1}^{k} \sum_{x \in C_{i,t-1}} ||x - \mu_{i,t-1}||^2$$

The above is exactly equal to the objective function from the previous iteration, $F(G, \mathbf{C}_{k,t-1})$. Therefore, the objective function never increases.

4 Other Clustering Objectives

We can alter the definition of centroids as follows:

$$\mu_i = \mu(C_i) = \arg\min_{\mu \in X} \sum_{x \in C_i} d(x, \mu)^2$$

Other than the k-means clustering objective, we can also use the $k-\mathit{medioid}$ objective function on a data set X:

$$F(G, \mathbf{C}_k) = \min \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i)^2$$

where each μ_i must be in N. Here, we're only looking at medioids within our data set X, unlike in k-means.

We can also use the k-median objective, defined as follows:

$$F(G, \mathbf{C}_k) = \min \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i)$$

The expression is the same as the k-medioid objective function and also requires that the μ_i be in X, but the distance function is no longer squared.

5 Clustering via Submodular Maximization

As discussed above, clustering can be described as the selection of a set of exemplars ("mediods") that best represent a large data set. These exemplars can be selected by solving the k-medioid problem which minimizes the sum of pairwise dissimilarities between exemplars and other points in the data set.

For a data set V, assume we are given a nonnegative function $d : V \times V \to \mathbb{R}$ such that $d(\cdot, \cdot)$ measures dissimilarity between elements. Then the cost function can be formulated as

$$L(S) = \frac{1}{|V|} \sum_{v \in V} \min_{e \in S} d(e, v)$$

In effect, we want to pick a set of points in S that, overall, minimize the distance that each node in V has to a point in S. Finding $S^* = \arg\min_{|S| \leq k} L(S)$ is NP-hard, but we can turn L into a monotone submodular function as follows:

$$f(S) = L(\{e_0\}) - L(S \cup \{e_0\})$$

where e_0 is some arbitrary "phantom" exemplar to begin with. From our earlier encounters with submodular functions, we know that we want to maximize a function rather than minimize it. So we can think about some sort of special point e_0 whose distance is really far away from all the other points—it's not in our data set, and it's way far off from everything else. So any set we choose has got to be an improvement upon the original set consisting of e_0. We then try to find the active set that, together with the phantom exemplar, reduces the value of our loss function more than any other set. Any e_0 such that $\max_{v' \in V} d(v, v') \leq d(v, e_0)$ for all $v \in V \setminus S$ is sufficient. Therefore, we can apply the approaches for optimizing submodular functions to the problem of clustering. We can use the greedy algorithm to get a $1 - 1/\epsilon$ approximation.

6 How do we connect this back to networks?

We can apply any clustering algorithm to provide us with a method for community detection in graphs.

Note that the distance functions we have considered used the Euclidean distance function $d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||$. If we apply k-means with the distance function $d(x, y) = BFS(x, y)$ such that $BFS(x, y)$ returns the shortest between nodes x and y returned by the BFS algorithm, the k-means algorithm will now approximate a clustering algorithm on a graph that detects communities within it. We can also consider weighted shortest paths, for instance. There is an art form in selecting the distance metric, but we can select it in a way that allows us to detect communities in graphs.

7 A High-Level View of Clustering (not covered in class, and therefore won't be tested, but it's pretty cool!)

What *is* clustering, actually? We can describe clustering at a high-level perspective as possessing the following properties (drawn directly from *Understanding Machine Learning*):

- **Scale Invariance (si)**: For any domain set X, dissimilarity function d, and any $\alpha > 0$, the following should hold: $F(X, d) = F(X, \alpha d)$ (where $\alpha d)(x, y) := \alpha d(x, y))$. The Scale Invariance requirement effectively communicates the requirement that a clustering function's results should not depend on the units used to measure between-point distances.

- **Richness (ri)**: For any finite X and every partition $C = (C_1, \ldots, C_k)$ of X (into nonempty subsets) there exists some dissimilarity function d over X such that $F(X, d) = C$. The Richness requirement asserts that the outcome of a clustering function should depend fully on the dissimilarity function d.

- **Consistency (co)**: If d and d' are dissimilarity functions over X such that for every $x, y \in X$, if x, y belong to the same cluster in $F(X, d)$ then $d'(x, y) \leq d(x, y)$ and if x, y belong to different clusters in $F(X, d)$, then $d'(x, y) \geq d(x, y)$, then $F(X, d) = F(X, d')$. This requirement refers to our basic intuition of clustering: we want similar points to be clustered together and dissimilar points to be separated; if points that already share a cluster become more similar, and points that are already separated become less similar, the clustering function should have even stronger "support" of its previous clustering decisions.

Given the above high-level definition of clustering, we have the following impossibility result:

Theorem. *There exists no function, F, that satisfies all there properties: Scale Invariance, Richness, and Consistency.*

For every pair of the three axioms, there exist clustering functions that satisfy the two properties in the pair, but no clustering function satisfies all three.